# **Angular Product Management Documentation**

## **Overview**

This Angular project allows users to view, add, and update products. The application includes:

* Fetch all products.
* Retrieve a single product by ID.
* Add a new product using a form.
* Update an existing product using a form.
* Display success messages upon product creation and update.

## **Project Structure**

product-management/

├── e2e/

├── node\_modules/

├── src/

│ ├── app/

│ │ ├── add-product/

│ │ │ ├── add-product.component.ts

│ │ │ ├── add-product.component.html

│ │ │ └── add-product.component.css

│ │ ├── update-product/

│ │ │ ├── update-product.component.ts

│ │ │ ├── update-product.component.html

│ │ │ └── update-product.component.css

│ │ ├── products/

│ │ │ ├── products.component.ts

│ │ │ ├── products.component.html

│ │ │ └── products.component.css

│ │ ├── app.component.ts

│ │ └── app.module.ts

└── app-routing.module.ts

│ ├── assets/

│ ├── environments/

│ └── environment.prod.ts   
 └── environment.ts   
└── index.html   
└── main.ts

├── angular.json

├── package.json

├── tsconfig.json

└── README.md

## **Components**

### **AddProductComponent**

This component allows users to add new products using a form.

#### **add-product.component.ts**

* **Properties**:
  + product: An object representing the new product.
  + successMessage: Displays a success message after adding the product.
* **Methods**:
  + addProduct(): Sends a POST request to the backend to add a new product. On success, a message is displayed to the user.

#### **add-product.component.html**

This file contains the form for adding a product. Key fields include:

* Product Name
* Product Price
* Units Available
* Product Description

#### **Example:**

<div class="add-product">

<h2>Add Product</h2>

<form (ngSubmit)="addProduct()">

<label for="name">Product Name:</label>

<input id="name" [(ngModel)]="product.name" name="name" required>

<label for="price">Product Price:</label>

<input id="price" [(ngModel)]="product.price" name="price" required>

<label for="units">Units Available:</label>

<input id="units" [(ngModel)]="product.units" name="units" required>

<label for="description">Product Description:</label>

<textarea id="description" [(ngModel)]="product.description" name="description"></textarea>

<button type="submit">Add Product</button>

</form>

<div \*ngIf="successMessage" class="success-message">{{ successMessage }}</div>

</div>

#### **Example POST API Response:**

* **POST /products**:

{

"message": "Product added successfully",

"product": {

"\_id": "671606ea74181402e6ec7a78",

"name": "New Product",

"description": "Product Description",

"price": 150,

"units": 20

}

}

### **ProductsComponent**

This component fetches and displays a list of all products.

#### **products.component.ts**

* **Properties**:
  + products: An array that holds all the products fetched from the backend.
* **Methods**:
  + ngOnInit(): This method is called when the component is initialized, and it fetches the list of products by calling productService.getProducts().

#### **products.component.html**

This file contains the structure for displaying the list of products in a table format. For each product, the product name, price, units, and description are shown.

#### **Example:**

<div class="products">

<h2>Products</h2>

<table>

<thead>

<tr>

<th>Name</th>

<th>Price</th>

<th>Units Available</th>

<th>Description</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let product of products">

<td>{{ product.name }}</td>

<td>{{ product.price }}</td>

<td>{{ product.units }}</td>

<td>{{ product.description }}</td>

</tr>

</tbody>

</table>

</div>

### **UpdateProductComponent**

This component is responsible for fetching a product by its ID and updating its details.

#### **update-product.component.ts**

* **Properties**:
  + product: Holds the details of the product being updated.
  + successMessage: Displays a success message after updating the product.
* **Methods**:
  + ngOnInit(): Retrieves the product by its ID using the productService when the component is initialized.
  + updateProduct(): Sends the updated product details to the backend and displays a success message.

#### **update-product.component.html**

This file defines the form used to update the product. Key elements include:

* Product Name
* Product Price
* Units Available
* Product Description

#### **Example:**

<div class="update-product">

<h2>Update Product</h2>

<form (ngSubmit)="updateProduct()">

<label for="name">Product Name:</label>

<input id="name" [(ngModel)]="product.name" name="name" required>

<label for="price">Product Price:</label>

<input id="price" [(ngModel)]="product.price" name="price" required>

<label for="units">Units Available:</label>

<input id="units" [(ngModel)]="product.units" name="units" required>

<label for="description">Product Description:</label>

<textarea id="description" [(ngModel)]="product.description" name="description"></textarea>

<button type="submit">Update Product</button>

</form>

<div \*ngIf="successMessage" class="success-message">{{ successMessage }}</div>

</div>

## **Services**

### **ProductService**

This service is used to interact with the backend API for all product-related operations.

* **Methods**:
  + getProducts(): Retrieves a list of all products.
  + getProductById(id: string): Retrieves a product by its ID.
  + addProduct(product: any): Sends a POST request to add a new product.
  + updateProduct(product: any): Sends a PUT request to update a product.

#### **Example:**

typescript

Copy code

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class ProductService {

private baseUrl = 'http://localhost:3000/products'; // Backend API URL

constructor(private http: HttpClient) { }

// Fetch all products

getProducts(): Observable<any[]> {

return this.http.get<any[]>(this.baseUrl);

}

// Fetch product by ID

getProductById(id: string): Observable<any> {

return this.http.get(`${this.baseUrl}/${id}`);

}

// Add a new product

addProduct(product: any): Observable<any> {

return this.http.post(this.baseUrl, product);

}

// Update product

updateProduct(product: any): Observable<any> {

return this.http.put(`${this.baseUrl}/${product.\_id}`, product);

}

}

## **Usage**

1. **Add Product**:
   * Navigate to the "Add Product" page.
   * Fill in the form fields for the product details.
   * Click **Add Product** to submit the form and create the product.
2. **View All Products**:
   * Navigate to the "Products" page to view a list of all products in a tabular format.
3. **Update Product**:
   * Navigate to the "Update Product" page by passing the product ID in the URL (e.g., /update-product/671606ea74181402e6ec7a78).
   * Edit the product details and click **Update Product** to submit the changes.

#### **Routing**

* **app-routing.module.ts** This file defines the routes for navigating between the add-product, update-product, and products components.

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { AddProductComponent } from './add-product/add-product.component';

import { UpdateProductComponent } from './update-product/update-product.component';

import { ProductsComponent } from './products/products.component';

const routes: Routes = [

{ path: 'add-product', component: AddProductComponent },

{ path: 'update-product/:id', component: UpdateProductComponent },

{ path: 'products', component: ProductsComponent },

{ path: '', redirectTo: '/products', pathMatch: 'full' },

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

#### **Usage**

1. **Add Product:**
   * Navigate to the "Add Product" page.
   * Fill in the form fields for the product details.
   * Click **Add Product** to submit the form and create the product.
2. **View All Products:**
   * Navigate to the "Products" page to view a list of all products in a tabular format.
3. **Update Product:**
   * Navigate to the "Update Product" page by passing the product ID in the URL (e.g., /update-product/671606ea74181402e6ec7a78).
   * Edit the product details and click **Update Product** to submit the changes.

]

#### **Requirements to Run the Project**

1. **Node.js** and **npm** must be installed.
   * Download and install from [Node.js official website](https://nodejs.org/).
2. **Angular CLI**:

Install the Angular CLI globally by running the following command:  
  
npm install -g @angular/cli

1. **Backend API**:
   * A backend API for managing products must be running, for example, an Express.js API with routes for product management.
2. **Dependencies**:

Install all dependencies by running:  
  
npm install

#### **Running the Project**

To run the project locally:

Start the Angular development server by running:  
  
ng serve

1. Navigate to http://localhost:4200 to view the application.